Database (DB) User

Assessment

INTEREST	TESTER	PROG OFFICE	ENG GROUP	USER	SECURITY	SYS ADMIN
TECHNICAL						
INFORMATIONAL	X	X	X	X		X
FUNCTIONAL			X			

Technical = requirements **Functional** = enhancements

Developer:	SAIC (on-site at the OSF)
Government POC:	MAJ James Quetsch
SME:	LT Kevin Brokaw (JSSC)
JITC Team:	Holly Lund and Sarah Patno

Assessment Objectives:
This was an assessment of the DB User functionality on GCCS V3.0 in the new operating environment (DII COE 3.0). There were no test procedures and the assessment application was not measured against requirements. JITC was present at all times and documented comments and prepared pre-assessment and post-assessment documentation. The user and JITC avoided all comments that may have caused redirection to the scope of the product. JITC was responsible to the Joint Staff and the PMO to bring all issues, questions, comments, and concerns to the appropriate GCCS representative.

User Expectations:
Observation of a system that functioned as well as the DB User currently resident in GCCS v2.2.

Assessment Results:
The overall assessment was excellent. All of the requirements were met, and the functionality was improved in the v3.0 as compared to v2.2.

User Comments:
The SME did have requests for improvements. The following are specific comments/concerns of the SME:

1. Integrate a graphic interface for easier user accessibility. Currently the application uses a DOS setup to access.
2. During the installation of any application, the choice to "delete icons" appears in a pop up menu. This requires the users to answer "yes" or "no" to each available icon. If "no" is answered the icon will be deleted, but than automatically reinstalled. This is only an inconvenience, but is an irritant to the user.
3. Improve the accessibility of the names to be granted/revoked. As it currently exists, all of the names are listed on the screen. The SME requests a more "user friendly" type of access to select names (since there is a list of approximately 1,200 names).
4. The SME expressed a desire to access the ability to grant/revoke privileges for PDR. At the time the application was assessed, PDR was not installed and SME was unable to access that capability. This capability might be a candidate for a test case during MDT&E Stage 3.

NOTE: *Future applications, which are "works in progress", will include the integration of a graphic interface and an improved ability to select users.*

User Information:

The following information is additional technical notes regarding the application provided by the developer:

“Introduction

In the current GCCS System various Database Segments will from time to time invoke ORACLE’s ‘sqlplus’ routine to execute a SQL script to accomplish some task or another. One of the features of sqlplus is the ability to specify the username and password to connect as on the command line that invokes the sqlplus application, thereby alleviating the need to embed a hardcoded username and password into the SQL script with the command ‘connect user/password’.

This, however, presents a problem that was recently determined when performed on a UNIX system. Specifically, when sqlplus is invoked with the command:

```
sqlplus user/password
```

That user name and password is visible to any user who performs a UNIX process list command, like:

```
ps -eaf
```

To prevent this from occurring the following method has been developed that proves effective at hiding the user name and password from the process line.

Kornshell Scripts

If you need to launch sqlplus from a kornshell script do it in this manner:

```
#!/bin/ksh

function Run_SQL {
su - oradba 1>/dev/null 2>/dev/null <<!
sqlplus <<EOF
$1
@$2 $3 $4
EOF
!
return $?
}

UNAME="USER/PASSWORD"
SQL_SCRIPT="junk.sql"
SQL_P1="MARK"
SQL_P2=""

Run_SQL $UNAME $SQL_SCRIPT $SQL_P1 $SQL_P2
print "Return Code is: $?"
```

Embed the ‘Run_SQL’ function in your kornshell script and then invoke as shown with whatever SQL script you choose. If you wish to have the user who runs the script to be something other than ‘oradba’ simply change the su line to suit.

The key to the function is the I/O redirection ‘<<EOF’ which will then read from the file lines to be fed until the ‘EOF’ line is reached. The first input is parameter 1 from the call, which happens to be the user name and password (UNAME), which is in answer to the first question sqlplus presents ‘Username:’ when it is invoked. This can be answered in a single line by having the user name and password on the same line separated by a slash., as is shown in the example.

C-Shell

Accomplishing this trick in C-Shell is similar, but a bit clumsier. See the example below:

```
#!/bin/csh

set UNAME = "USER/PASSWORD"
set SQL = "junk.sql"
set SQL_P1 = "MARK"
set SQL_P2 = ""

su - oradba >&/dev/null <<!
sqlplus <<EOF
${UNAME}
@${SQL} ${SQL_P1} ${SQL_P2}
EOF
!
echo “Return Code is: $?”
```

The trick is the same, redirect the input to retrieve the user name and password from the file, but the problem is the C-Shell does not allow the creation of subroutines to be more modular. Therefore, every time you wish to invoke a sqlplus script you will need to embed this sequence in your C-Shell routine.

PERL

If you are a PERL programmer the procedure becomes a bit more elegant:

```
#!/h/COTS/PERL/progs/perl

$UNAME = "USER/PASSWORD";
$SQL = "junk.sql";
$SQL_P1 = “MARK”;
$SQL_P2 = “”;

$run_cmd = "su - oradba -c 'sqlplus <<EOF \n$UNAME\n\@$SQL $SQL_P1
$SQL_P2\nEOF'";

$return_code = system ("$run_cmd");
print “Return Code is: $return_code”;
```

Here the entire ‘su’ command with the sqlplus command are embedded in a single line (the break is only due to the length of the line). Please note the use of the ‘\n’ which is interpreted as new lines, when the command is passed to the PERL ‘system’ call. The other important factor is the use of the single quotes, just before the sqlplus and just after the second EOF, which denotes the limits of the command to be executed inside the super-user invocation.

Sample SQL Script

The SQL script that is invoked with these examples is shown below:

```
select user from dual;
select * from all_tables where owner='&1';
exit 2;
```

This shows how to use a parameter inside the script, that is passed in. &1 for the first, &2 for the second and so on. It also shows how to exit from the SQL script with a status code other than Zero, by using the 'exit <value>' option. Provided you use this example SQL script, every one of the shell scripts provided should at the end display:

```
Return Code is: 2
```

Common Factors

For each of these examples there is a common thread of use to provide a better way to compare them.

1) Each performs a UNIX super-user 'su' command to the 'oradba' account prior to invoking sqlplus. If the developer wishes to connect to a different account, that is possible, as long as that account has access to the sqlplus executable.
2) The use of 'USER/PASSWORD' is simply for example purposes. There probably is no ORACLE account called 'USER', and it certainly does not have the password of 'PASSWORD'. Please substitute the appropriate account to connect to, so that your SQL script will operate correctly.
3) It is possible to use EXTERNAL Password verification with this methodology, simply use a single slash (/) character for the UNAME. Provided that the Super-User connection is to an account that has the ORACLE account identified EXTERNALLY, it will work correctly.

Conclusion

There are probably other ways to accomplish the same goal as described above, but these have been tested and seem to address the primary issue. These mechanisms are not that difficult to use, once they are understood and they provide no additional limitations on capabilities or performance. For other Database Systems there will possibly need to be some modifications, but nothing that should be difficult to adjust to."

Recommendations:
The SME felt that the application should be integrated into GCCS v3.0.